**Actas XVIII JENUI 2012, Ciudad Real, 10-13 de julio 2012**
**I.S.B.N. 10: 84-615-7157-6 | I.S.B.N. 13:978-84-615-7157-4**
**Páginas 335-342**

# La Reingeniería como tópico en la Docencia de la Ingeniería del Software: una Experiencia Práctica

Ricardo Pérez-Castillo, Félix García, Ignacio García Rodriguez de Guzmán y Mario Piattini
Instituto de Tecnologías y Sistemas de Información (ITSI)
Universidad de Castilla-La Mancha
Paseo de la Universidad 4, 13071, Ciudad Real, España
[ricardo.pdelcastillo, ignacio.grodriguez, felix.garcia, mario.piattini]@uclm.es

## Resumen

El mantenimiento software es reconocido como una de las áreas de conocimiento más relevantes dentro de los currículos internacionales de ingeniería del software. A pesar de ello y de su importancia en la industria, no es usual realizar prácticas docentes sobre técnicas de mantenimiento software como es la reingeniería. Este artículo presenta una experiencia docente en la que se realiza una práctica de laboratorio sobre reingeniería utilizando herramientas de ingeniería inversa y generación de código. El proceso de enseñanza aprendizaje es evaluado cualitativa y cuantitativamente mediante el contraste de resultados entre una evaluación inicial y final. Como resultado se observó que el alumnado desconocía la reingeniería como una técnica de mantenimiento software y su satisfacción con la experiencia fue alta o muy alta (65%) o media (35%). Como lecciones aprendidas destacar que el alumnado señaló el manejo de herramientas de reingeniería como muy útil para su desempeño profesional futuro y la necesidad de dedicar más tiempo en el aula para el aprendizaje de dichas herramientas.

## Summary

Software maintenance is recognized as an important knowledge area within the most common international curriculums in software engineering. Despite this fact, and its importance in the industry, software maintenance and supporting techniques such as reengineering are hardly ever taught in practical lessons. This paper presents a reengineering teaching experience conducted in lab sessions by using reverse engineering and code generation tools. The teaching-learning process is qualitative and quantitatively assessed by comparing results between an initial and final evaluation. Reported results show that students do not know reengineering as a software maintenance technique and their satisfaction with the experience was high or very high (65%) or medium (35%). The key learned lessons are that students recognized the usage of reengineering tools as very convenient for their performance as future practitioners and the need to devote additional time in classroom to learn such tools.

## Palabras Clave

Ingeniería del Software, Mantenimiento, Reingeniería, Práctica, Evaluación.

## 1. Introducción

Dentro del proceso de desarrollo software, el mantenimiento es una de las fases que más esfuerzo y recursos conlleva [8]. De hecho, el esfuerzo en mantenimiento está entre un 70 y 80% frente al 20 y 30% del resto de fases del ciclo de vida del desarrollo software [6, 10].

El mantenimiento software es una actividad clave para corregir, adaptar, migrar o mejorar un sistema software [5]. El mantenimiento, por un lado, favorece un incremento en la satisfacción del usuario final, y por otro, alarga la vida de los sistemas software, lo cual mejora el retorno de la inversión.

Para llevar a cabo el mantenimiento del software existen diversos enfoques y numerosas técnicas. Uno de los enfoques que se ha aplicado con más éxito durante las últimas décadas es la reingeniería [1]. La reingeniería aboga por obtener versiones mejoradas de un sistema software primando la reutilización con el fin de preservar la información de negocio embebida en el sistema bajo mantenimiento [9]. La reingeniería consiste en tres fases [2]:

Ingeniería inversa, la cual analiza los elementos de un sistema existente y sus interrelaciones y obtiene una representación abstracta del sistema.
Reestructuración: la cual introduce, sobre la representación abstracta, alguna mejora de diseño o nueva funcionalidad.
Ingeniería directa: la cual genera la implementación física del nuevo sistema software.
Este enfoque también se denomina modelo de reingeniería en herradura [7], debido al cambio del nivel de abstracción a lo largo de las tres fases (véase Figura 1).

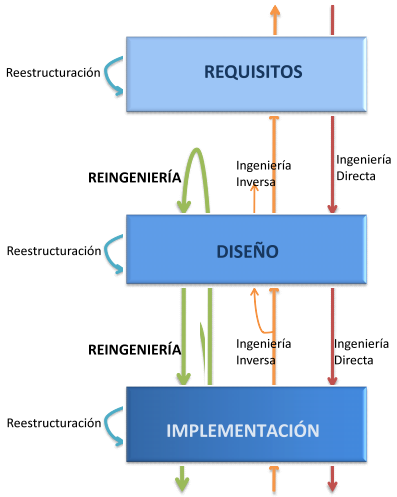


Figura 1. Modelo de reingeniería en herradura.

A nivel académico, el mantenimiento software es también percibido como un tema de interés. De hecho, la reingeniería y en general el mantenimiento software son incluidos en los currículos docentes internacionales. En el currículo del SWEBOK (*Software Engineering Body of Knowledge*) [4], el sexto capítulo es dedicado completamente al mantenimiento software. También el *Software Engineering Curriculum* propuesto por ACM e IEEE [3], establece una área de conocimiento para evolución software con una duración estimada de 10 horas (un 2.24% del tiempo total).

A pesar de la relevancia del mantenimiento software tanto a nivel industrial como académico, se piensa que la docencia sobre mantenimiento software adolece de dos problemas clave:

1. Dentro de las universidades se favorece una visión del mantenimiento como una fase rutinaria y ajena al resto de fases del desarrollo software.
2. No existe mucha investigación acerca de la docencia sobre mantenimiento software en comparación con el desarrollo software.

Con el fin de contribuir a la solución de esos problemas, este artículo presenta una experiencia docente sobre reingeniería software. La experiencia consistió en la impartición de un seminario sobre reingeniería y mantenimiento en clase de teoría, y posteriormente una práctica grupal de reingeniería. El objetivo principal fue analizar si la experiencia docente propuesta es pedagógica y mejora el proceso enseñanza-aprendizaje sobre reingeniería, y en general sobre mantenimiento software. Para ello, se realizó al alumnado una evaluación inicial y otra final mediante test. Los tests permitieron contrastar por un lado lo aprendido sobre reingeniería, y por otro, las dificultades de aprendizaje más comunes que presentaba el alumnado.

Entre las lecciones aprendidas tras esta experiencia docente destaca como una de las dificultades que el alumnado encontró la integración de nuevas funcionalidades durante la reingeniería. En cualquier caso, el alumnado valoró positivamente el aprendizaje y manejo de nuevas aplicaciones para soportar la reingeniería software.

El resto del artículo se organiza de la forma siguiente. La Sección 2 introduce los conceptos principales de mantenimiento y reingeniería software a la vez que los contextualiza en los currículos académicos de referencia. La Sección 3 presenta en detalle la experiencia docente sobre reingeniería llevada a cabo y en la Sección 4 se analizan los resultados obtenidos. Finalmente, la Sección 5 presenta las conclusiones y lecciones aprendidas de este trabajo.

## 2. Reingeniería software y docencia

En la actualidad, existen diversos curriculos nacionales e internacionales enfocados a la profesión de la Ingeniería del Software, como son SWEBOK, Computing Curricula, ICF-2000 (de IFIP/UNESCO), ISCC, entre otros. Sin embargo, SWEBOK e ACM/IEEE SE son probablemente

las más relevantes. En esta sección se presentará cómo estos curriculos abordan el proceso de mantenimiento y más concretamente, la reingeniería como herramienta fundamental para este proceso.

SWEBOOK (Software Enginegin Body of Knowledge) que se concibió para la acreditación de los currículos universitarios y la certificación de profesionales, identifica un cuerpo básico de conocimiento que caracteriza la disciplina de Ingeniería del Software. SWEBOK está dividido en 10 áreas de conocimiento, entre las que está incluído el Mantenimiento del Software. Dentro del mantenimiento, SWEBOK considera fundamentos, conceptos clave, el procesos y técnicas para el mantenimiento (ver **Figura 2**). Tal y como se observa en la **Figura 2**, las tres técnicas están englobadas en el concepto de la reingeniería (técnica central), ya que tanto la comprensión como la ingeniería inversa forman parte de la reingeniería.

El *Computing Curricula* de ACM/IEEE-CS proporciona guías para la elaboración de planes de estudio de las carreras de ciencias de la computación y de ingeniería informática. El *Computer Curricula* está compuesto por diversas partes: un volumen general y volúmenes adicionales destinados a las disciplinas específicas. Uno de estos volúmenes, el SE2004, se enfoca en la descripción de la Ingeniería del Software. Es importante destacar que de entre todas las áreas de conocimiento (o tópicos de interés que son transversales a todas las disciplinas) contemplados en el *Computing Curricula*, la disciplina de Ingeniería del Software descrita en el SE2004 es la que más esfuerzo estima para abordar el área de Mantenimiento del Software.

El SE2004 se estructura en un total de 12 unidades de área, una de las cuales está directamente relacionada con el mantenimiento y la reingeniería. La unidad de área SE7, titulada "Evolución del Software", contempla el mantenimiento del software, las características del mantenimiento del software, la reingeniería, los sistemas heredados y la reutilización del software.

Los dos curriculos brevemente presentados muestran cómo el proceso de mantenimiento recibe tanta importancia como otros procesos estudiados en la Ingeniería del Software, y concretamente, ambos incluyen la reingeniería como herramienta principal para el desarrollo del mantenimiento. Sin embargo, observando los planes de estudio actuales puede observarse que en la mayoría de los casos: (i) el mantenimiento se estudia brevemente y a nivel teórico, y (ii) la reingeniería apenas recibe alguna referencia, pudiendo incluso llegar a no ser mencionada en la mayoría de las asignaturas relacionadas.

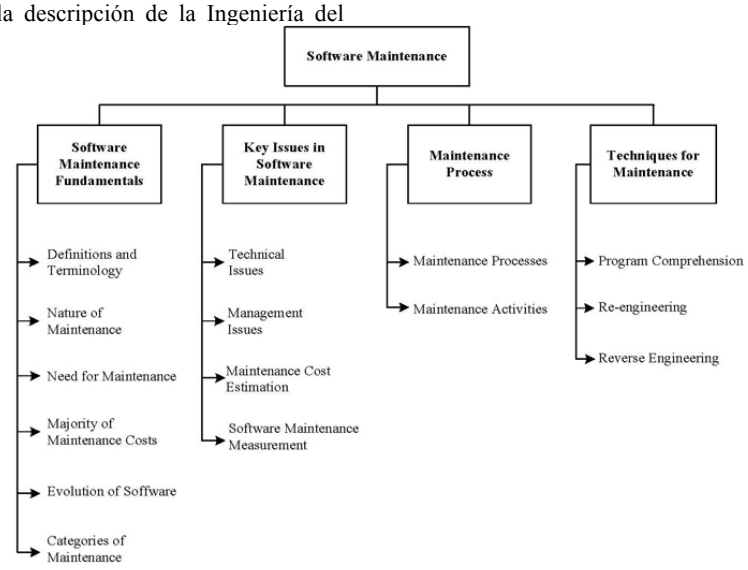


Figura 2. Área de conocimiento de Mantenimiento del Software

## 3. Práctica propuesta

Esta sección presenta en detalle la experiencia docente sobre reingeniería. En primer lugar se presenta el contexto de la experiencia docente. En segundo lugar se describe la práctica llevada a cabo, mostrando en tercer lugar su ejecución.

### 3.1. Contexto

La experiencia docente tuvo lugar durante el primer cuatrimestre del curso 2011/2012 en la asignatura de "Ingeniería del Software" correspondiente a las titulaciones de Ingenierías Técnicas en Informática de Gestión (ITIG) e Informática de Sistemas (ITIS) de la Escuela Superior de Informática (ESI) de Ciudad Real. Se trata de una asignatura anual de 10 créditos ECTS de 3º curso de la titulación. La Tabla 1 muestra los contenidos teóricos y de laboratorio impartidos en la asignatura.

| **TEORÍA (70%)** | |
|---|---|
| I. Fundamentos de la I. Software (25%) | 1. Introducción a los Sistemas de Información |
| | 2. Ciclo de Vida del software |
| | 3. Metodologías de desarrollo de software |
| | 4. Inicio de un proyecto y gestión de requisitos software |
| II. Desarrollo Estructurado (25%) | 5. Análisis y Diseño |
| | 6. Pruebas Software |
| III. Desarrollo Orientado a Objetos (50%) | 7. Conceptos básicos de OO |
| | 8. El Leng. Unif. Modelado (UML) |
| | 9. Metodologías OO |
| | 10. La metodología MÉTRICA v3 |
| | 11. Proceso Unificado Desarrollo |
| | 12. Casos de Uso |
| | 13. Modelado Estructural |
| | 14. Diagramas de Interacción |
| | 15. Modelado del Comportamiento |
| | 16. Modelado Arquitectónico |
| | 17. Pruebas en OO |
| | 6. Pruebas Software |
| **LABORATORIO (30%)** | |
| 1. JDBC (12,5%) | |
| 2. Análisis y Diseño Est. con Easy Case (25%) | |
| 3. Pruebas Unitarias con JUnit (12,5%) | |
| 4. Desarrollo OO con Visual Paradigm (50%) | |

**Tabla 1.** Contenidos de Ingeniería del Software

La asignatura se imparte según las directrices del EEES y en ella se encuentran matriculados 34 alumnos de ITIG y 46 alumnos de ITIS.

### 3.2. Descripción de la práctica

La actividad docente consistió en la impartición de un seminario teórico en el aula de clase de una hora de duración en el que se explicaron los fundamentos de la Reingeniería y su relación y contribución al campo del Mantenimiento del Software y de una sesión de laboratorio de dos horas de duración en la que se explicó el uso de las herramientas de reingeniería a utilizar y en la que los alumnos realizaron y entregaron la práctica (véase Figura 3).

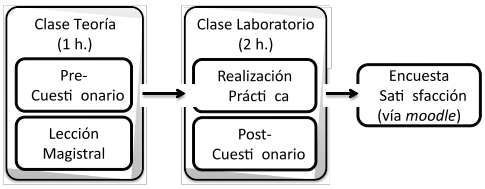


Figura 3. Vista general de la experiencia docente

La práctica consistía en, a partir de cinco ficheros ejecutables (.class) de un sistema de información obsoleto sin documentación asociada, obtener un sistema mejorado a partir de nuevos requisitos que se debían satisfacer modelando con UML y programando en JAVA (véase apéndice II).

Con el fin de recoger la realimentación necesaria para evaluar esta iniciativa se diseñaron tres cuestionarios:

**Cuestionario *Pre-***, que fue rellenado por los alumnos antes del comienzo del seminario teórico y su objetivo era evaluar los conocimientos previos de los alumnos en el campo de reingeniería considerando las asignaturas previas que habían superado. Este cuestionario tipo test constó de tres preguntas conceptuales y dos casos prácticos sobre reingeniería donde tenían que seleccionar la mejor toma de decisiones (véase apéndice I).

**Cuestionario *Post-***, que fue rellenado a la finalización de la práctica y que incluía preguntas tipo test sobre el tema de reingeniería, con las que evaluar el conocimiento adquirido por los alumnos como resultado de la actividad. Este cuestionario tipo test se realizó vía *moodle* con cinco preguntas conceptuales sobre mantenimiento y reingeniería y los mismos casos prácticos del cuestionario inicial.

**Cuestionario Final** en el que se pedía la opinión vía *moodle* a los alumnos sobre si incluirían la actividad en los contenidos de la asignatura así como los aspectos que consideraban positivos y negativos de la experiencia.

El desarrollo de la actividad tuvo lugar sin contratiempos, participaron un total de 68 alumnos (29 ITIG, 39 ITIS) lo que supuso una

participación del 85% de los matriculados. El material entregado fue completo en todos los casos por lo que no se tuvo que descartar a ningún alumno en el estudio. Los resultados obtenidos se analizan en el siguiente apartado.

## 4. Resultados

Esta sección presenta un análisis cualitativo y cuantitativo de los resultados obtenidos en la experiencia docente explicada. La Sección 4.1 analiza los resultados de la evaluación inicial. La Sección 4.2 estudia los resultados obtenidos en la práctica de laboratorio. La Sección 4.3 examina los resultados obtenidos en la evaluación final presentando una comparativa entre las evaluaciones iniciales y finales. Finalmente, la Sección 4.4 recoge la opinión del alumnado acerca de la experiencia docente.

### 4.1. Evaluación inicial

En las preguntas conceptuales sobre los fundamentos del mantenimiento y la reingeniería se observó que más del 70% del alumnado tenía una ligera o bien formada idea acerca de que es el mantenimiento software. Sin embargo, tan sólo un 22% del alumnado tenía una idea vaga acerca de que es la reingeniería. Además, tan sólo un 17% supo establecer algún tipo de relación entre mantenimiento software y reingeniería. Estos resultados mostraban claramente que el alumnado desconocía la reingeniería como técnica para mantener sistemas software, lo cual justifica la experiencia docente sobre esta temática.

En cuanto a los casos prácticos (véase apéndice I) se obtuvieron mejores resultados con un 62% y 66% de acierto en los dos casos prácticos respectivamente. No obstante, se puede considerar un porcentaje de acierto discreto.

### 4.2. Práctica de laboratorio

La Tabla 2 muestra la calificación obtenida por cada uno de los grupos en la práctica de laboratorio así como los criterios de evaluación no alcanzados. La media de la calificación para todo el alumnado fue 8.18 con una desviación típica de 0.9.

Las calificaciones obtenidas muestran que el alumnado no encontró grandes dificultades para abordar la práctica propuesta. A pesar de ello, se detectó una serie de errores que el alumnado realizó de forma sistemática. La Tabla 3 muestra los criterios de evaluación no alcanzados así como su frecuencia en el alumnado durante la experiencia docente.

| Grupo | Calific. [0-10] | Errores | | | | |
|---|---|---|---|---|---|---|
| | | C1 | C2 | C3 | C4 | C5 |
| 1 | 9 | | ♦ | | | |
| 2 | 8 | | ♦ | ♦ | | |
| 3 | 9 | | ♦ | | | |
| 4 | 8 | | ♦ | ♦ | ♦ | |
| 5 | 7.5 | ♦ | ♦ | | ♦ | |
| 6 | 8.5 | | ♦ | ♦ | | |
| 7 | 9 | | ♦ | | | |
| 8 | 8 | ♦ | ♦ | | | |
| 9 | 9 | | ♦ | | | |
| 10 | 7.5 | ♦ | ♦ | ♦ | | |
| 11 | 8.5 | | ♦ | | ♦ | |
| 12 | 7.5 | ♦ | ♦ | ♦ | | |
| 13 | 6 | ♦ | ♦ | ♦ | ♦ | ♦ |
| 14 | 9 | ♦ | | | | |

**Tabla 2.** Calificaciones grupales de la práctica

| ID | Criterio de evaluación no alcanzado | % |
|---|---|---|
| C1 | La clase *Cliente* no está bien integrada (faltan asociaciones) | 40.6 |
| C2 | Le falta dependencias con la nueva clase Cliente | 89.9 |
| C3 | No funciona el *Launcher* para probar el sistema | 42.0 |
| C4 | Confusión en las fases de la reingeniería en la memoria | 30.4 |
| C5 | Errores en el código generado / de compilación. | 7.2 |

**Tabla 3.** Errores en la práctica y su frecuencia

Por un lado el error más repetido consistió en una integración incorrecta de la nueva clase *Cliente* que debían añadir al sistema en la fase de reestructuración. Esto fue debido a la ausencia de dependencias con otras clases (89.9%) así como a la ausencia de asociaciones (40.6%), es decir, no se modificó el tipo de las variables que represaban a un cliente.

Por otra parte, aunque en menor medida, se detectaron errores de compilación (criterio C5) y en el código fuente, sobre todo de la clase principal que debían implementar (criterio C3).

Finalmente, también hay que destacar que el 30.4% del alumnado contextualizó alguno de los pasos y tareas realizados en una fase equivocada de la reingeniería (véase el criterio C4 en la Tabla 3).

### 4.3. Evaluación final

Los resultados obtenidos en la evaluación final con el post-cuestionario mostraron que la percepción del alumnado sobre reingeniería había cambiado. Un 90% de los alumnados supieron seleccionar la definición correcta de reingeniería lo que supuso un incremento del 68% del alumnado. Por otra parte, más del 79% relacionaron adecuadamente reingeniería con mantenimiento, lo cual supuso un incremento del 62% frente a los resultados obtenidos en la evaluación inicial.

Además, se calificó cada pre-, post-cuestionario del 0 al 10, asignando una nota a cada alumna o alumno. Se observó que la media de puntuación obtenida en el post-cuestionario fue de 6.7, mientras que la obtenida en el pre-cuestionario fue de 1.9. La Figura 4 muestra la distribución de notas obtenidas en los dos cuestionarios. La nota del alumnado en ambos casos sigue una distribución normal de medias 1.9 y 6.7, y una desviación típica de 1.9 y 2.6. Como se observa, la incidencia de la práctica docente sobre reingeniería aumentó el conocimiento del alumnado sobre reingeniería y mantenimiento. No obstante, la efectividad del proceso enseñanza/aprendizaje fue desigual, ya que la distribución de notas en el post-cuestionario muestra una desviación típica superior a la obtenida tras la evaluación inicial.

Por otra parte se evaluó la diferencia entre los resultados obtenidos en los dos casos prácticos propuestos en ambas evaluaciones (véase apéndice I). Para ello se realizó una prueba T de *Student* sobre los resultados obtenidos en ambos cuestionarios para cada uno de los dos casos prácticos (véase Tabla 4). Para los dos casos prácticos existe una diferencia de medias entre la evaluación inicial y final con un grado de significación del 99%. La diferencia de medias es la misma en los dos casos prácticos (-0.21). Una diferencia negativa significa que la media de puntuación obtenida en el post-cuestionario es más alta que la puntuación obtenida al inicio de la experiencia docente. Además, si se observa el tamaño del efecto, se observa que el alumnado mejora más en el segundo caso práctico (-1.03) que en el primero (-0.75).

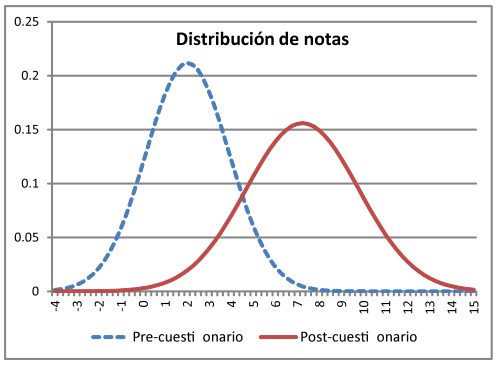


Figura 4. Distribución de notas obtenidas en la evaluación inicial y final

| | Dif. Media | Des. Tip. | T-Student | Tamaño Efecto | Sig. |
|---|---|---|---|---|---|
| C I | -0.21 | 0.57 | -3.03 | -0.75 | 0.004 |
| C II | -0.21 | 0.41 | -4.18 | -1.03 | 0.000 |

**Tabla 4.** Comparación de casos prácticos

### 4.4. Satisfacción del alumnado

Al finalizar la experiencia docente, se distribuyó un cuestionario para conocer la opinión del alumnado. A la pregunta de si incluirían el mantenimiento y reingeniería en la asignatura, más del 51% indicaron que lo incluirían tanto en teoría y en prácticas. Por otra parte, más del 45% lo incluirían en prácticas.

Sobre la pregunta para recoger su satisfacción de la experiencia docente sobre reingeniería, la cual podía ser valorada en una escala del 1 al 5, se obtuvieron los siguientes datos: el 60% tuvo una satisfacción alta (4); el 35% una satisfacción media (3); el 2% muy alta (5).

En el cuestionario también se recogió la opinión del alumnado en forma de puntos positivos y negativos. Como puntos positivos, la mayoría destacó que (i) aprendieron a reutilizar código y que no siempre hay por que partir desde cero para crear un sistema software; y (ii) valoraron la utilidad de conocer y utilizar nuevas herramientas de reingeniería, sobre todo las relacionadas con la ingeniería inversa como los descompiladores, que nunca antes habían usado. Finalmente, al alumnado casi unánimemente señaló como aspectos negativos la escasez de tiempo para realizar la práctica de laboratorio y una explicación reducida de las herramientas de reingeniería a utilizar.

## 5. Conclusiones

A la hora de abordar la docencia en Ingeniería del Software, es imprescindible capacitar a los alumnos para desarrollar su trabajo en los entornos profesionales actuales, en los que se les demanda la realización de tareas que no siempre cubren los planes de estudio. Un gran número de ingenieros trabaja día a día en el mantenimiento del software aplicando reingeniería, pero este proceso suele recibir poca atención en cuanto a la docencia del mismo.

En este artículo, se presenta una experiencia llevada a cabo en una asignatura de Ingeniería del Software para alumnos de último curso. Como resultado se han recogido evidencias iniciales sobre: (1) la falta de conocimiento que el alumnado presenta sobre el mantenimiento del software y reingeniería (alumnado que además está cercano a incorporarse al mundo laboral), y (2) cómo dicho alumnado puede suplir de forma considerable este vacío mediante una práctica especialmente diseñada para adquirir la formación básica necesaria para enfrentarse al mantenimiento del software aplicando reingeniería.

Los resultados positivos obtenidos con esta experiencia piloto serán de gran ayuda para introducir los temas relacionados con el mantenimiento y la reingeniería en las nuevas asignaturas de grado de Ingeniería del Software en la ESI Ciudad Real, que entran en vigor a partir de los cursos académicos siguientes.

### 5.1. Lecciones aprendidas

Entre las lecciones aprendidas que pueden ser aplicadas a futuras repeticiones de la experiencia docente destacan las siguientes:

1. En la realización de la práctica, la mayor dificultad que encontró el alumnado fue la introducción de nuevas funcionalidades en el sistema destino. Sobre todo tuvieron problemas para integrar las nuevas funcionalidades con las existentes durante la fase de reestructuración.
2. Uno de los errores más comunes que cometió el alumnado fue la confusión entre términos y las diferentes fases de la reingeniería, ya que no tenían muy claro las fronteras entre ingeniería inversa, reestructuración e ingeniería directa. Como consecuencia, se plantea el incidir en esta parte en la clase magistral inicial en futuras repeticiones.
3. El alumnado señaló como un hándicap en la realización de la práctica el escaso tiempo con el que contaron (2 horas). Por lo tanto, se plantea extender el tiempo de entrega en futuras repeticiones.
4. El alumnado encontró de gran utilidad conocer y usar nuevas herramientas como descompiladores y otras aplicaciones de ingeniería inversa para generar modelos UML. No obstante, el alumnado indicó la necesidad de dedicar más tiempo a la explicación del manejo de estas herramientas. En futuras repeticiones de la experiencia se plantea el hecho de impartir una clase magistral práctica sobre el manejo de estas herramientas.

## Agradecimientos

Este trabajo ha sido financiado por el programa FPU (Formación de Profesorado Universitario) de la Secretaría de Investigación.

## Referencias

[1] Bianchi, A., D. Caivano, V. Marengo, y G. Visaggio, Iterative Reengineering of Legacy Systems. IEEE Trans. Softw. Eng., 2003. **29**(3): p. 225-241.

[2] Chikofsky, E.J. y J.H. Cross, Reverse Engineering and Design Recovery: A Taxonomy. IEEE Softw., 1990. **7**(1): p. 13-17.

[3] Computing Curriculum Project, Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering (SE2004). http://sites.computer.org/ccse/, 2004, IEEE Computer Society & ACM.

[4] IEEE Computer Society, Guide to the Software Engineering Body of Knowledge (SWEBOK). http://www.computer.org/portal/web/swebok, 2004.

[5] ISO/IEC, ISO/IEC 14764:2006. Software Engineering -- Software Life Cycle Processes -- Maintenance. http://www.iso.org/iso/catalogue_detail.htm?csnumber=39064, 2006, ISO/IEC.

[6] ISO/IEC, ISO/IEC 12207:2008 - Systems and software engineering -- Software life cycle processes, 2008.
[7] Kazman, R., S.G. Woods, y S.J. Carrière. Requirements for Integrating Software Architecture and Reengineering Models: CORUM II. in Proceedings of the Working Conference on Reverse Engineering (WCRE'98). 1998. IEEE Computer Society.
[8] Lehman, M.M., On understanding laws, evolution, and conservation in the large-program life cycle. Journal of Systems and Software, 1979. **1**: p. 213-221.
[9] Sneed, H.M., Planning the Reengineering of Legacy Systems. IEEE Softw., 1995. **12**(1): p. 24-34.
[10] Sneed, H.M., Estimating the Costs of a Reengineering Project. Proceedings of the 12th Working Conference on Reverse Engineering2005: IEEE Computer Society. 111 - 119.

Apéndice I. Casos prácticos de los cuestionarios

CASO I. Se tiene un sistema construido mediante un desarrollo estructurado (escrito en C) y se quiere migrar a un sistema desarrollado mediante orientación a objetos (escrito en Java).

a) No haría nada. Si el sistema funciona para que cambiarlo a Java. La migración tendría un coste asociado muy alto.
b) Ya que el cliente lo desea, pondría a varios programadores expertos en C y Java a programar el sistema nuevo.
c) Pondría a varios programadores expertos en C y Java a programar el sistema nuevo. Para futuros mantenimientos, un desarrollo orientado a objetos será más mantenible (debido entre otras cosas a herencia, polimorfismo, etc.) y por lo tanto aunque ahora se invierta dinero, los siguientes mantenimientos serán más baratos.
d) No es necesario migrar. Es mejor recoger con el cliente una especificación de requisitos actual (ya que puede haber nuevos requisitos) y crear un sistema Java desde cero. Así no gastamos tiempo en entender el código C.

CASO II. Se dispone de un sistema de gestión de ventas que guarda toda la información de clientes y productos en ficheros de texto plano. Por lo que se plantea que el sistema guarde la información en una base de datos relacional, más adecuado al mercado actual.

a) Si el sistema se conecta con una base de datos, ciertas funcionalidades podrían dejar de funcionar correctamente en el sistema. Es mejor crear un sistema nuevo que haga lo mismo que el anterior y que ya guarde la información en una base de datos.
b) Se crearán la bases de datos y la única modificación en el sistema será crear un agente de bases de datos que centralizará el acceso a datos del sistema existente.
c) Si el sistema funciona con ficheros no hay porque integrarlo con bases de datos. Ya se realizará en el futuro cuando se cree un nuevo sistema para el cliente.

## Apéndice II. Enunciados de la práctica

Para esta segunda práctica, se llevará a cabo un proyecto de reingeniería. Para ello cada grupo de trabajo tomarán como artefactos software de origen el conjunto de ficheros ejecutables (*.class) pertenecientes a un sistema de información obsoleto del cual no se tiene nada de documentación adicional y su funcionalidad a priori es desconocida. El objetivo es obtener un sistema mejorado (y documentado mediante diagramas UML) en el cual se introducirán los siguientes cambios:

Añadir la clase Cliente (con NIF, nombre y apellidos) la cual será utilizada en aquellos casos que una variable de tipo String represente un cliente.

Desarrollar una clase Main que permita ejecutar las principales funcionalidades del sistema mejorado.

Se deberá seguir y documentar cada una de las tres fases de la reingeniería (ingeniería inversa, reestructuración, e ingeniería directa) para obtener el SI mejorado. Como resultado de la práctica, el grupo entregará un informe en el que se describan los pasos seguidos en cada fase indicando los problemas encontrados. Además, se listarán los artefactos software obtenidos en cada una de las fases.